High Performance Computing Systems Monitoring Implementation

Cindy Martin, Jeff Johnson, HPC-3

Making efficient use of data is increasingly critical to the maintainability of HPC systems as they continue to grow in scale. Previous methods used by HPC Division manipulated the same data separately for different functions or entirely separate data streams, so that there was no cohesive view of an HPC environment. The HPC monitoring project charter was to create a tool that could enable a holistic view and in turn increase efficiency. By leveraging data management for each data-centric function necessary in an HPC environment, all of them can become more efficient and scalable.

As a starting point, the HPC monitoring project selected Zenoss, which provided a well-documented and cohesive code base with a rich user interface framework. However, the provided core was not able to handle the scale of data being generated in a high-performance computing environment, and there were several deficiencies that needed to be addressed. Areas that required local enhancement were data import speed, model history storage, issue tracking, asset tracking, event-driven commands, and job data correlation (Fig. 1).

The ability to scale is critical to the success of any HPC monitoring system. The project conducted several load test scenarios using the Zenoss core product. The core application could handle approximately 500 messages per second—our need was much greater, approximately 1,500 messages per second. The HPC implementation addressed this requirement by creating a hierarchy of Zenoss monitoring instances.

A new interface was integrated into the Zenoss core product, providing a seamless transition between multiple, tiered monitoring instances. Figure 2 depicts the hierarchical implementation. The interface handles the redirection to the appropriate Zenoss instance based on the model hierarchy defined in the master monitoring Zenoss instance. In this configuration, the lowest tier collects events from the connected units within the cluster and the cluster tier filters and aggregates from the node tier. The master instance, in turn, filters and aggregates from the lower tiers.

Another requirement was the ability to mitigate user impact by automatically removing nodes from service, based on system events. This functionality was added to the core Zenoss product by implementing event driven commands, which place nodes offline based on hardware events or predefined thresholds. Notification of the action is then relayed to operations staff.

Problems may be reintroduced to a cluster when previously repaired components are placed back into service. Asset and issue tracking are essential to the identification of these problems. The Zenoss core was enhanced to track the repairs done to hardware and the location history within the cluster. Issue tracking also facilitates the automated calculation of downtime necessary for availability reporting.

The local enhancements to the Zenoss core facilitate a more comprehensive view of job and system health. Toward this end the project created an interface to correlate job events with system events (Fig. 3). This interface brings together the Moab resource scheduler event log data with the system event data collected in Zenoss. Currently, we are working with a subset of HPC system users to expand and improve this functionality.

Additional data sets will be added, including a more comprehensive set of environmental data associated with components within the cluster. In addition, the temperature and voltage data associated with the room should be included within the next year.

The HPC Zenoss monitoring tool has been deployed within all LANL high-performance-computing enclaves. HPC-3 has diligently shared LANL enhancements with the Zenoss open source community. The HPC organization anticipates that the use of this monitoring tool will enhance the stability and robustness of our high performance systems.

For more information contact Cindy Martin at c_martin@lanl.gov.

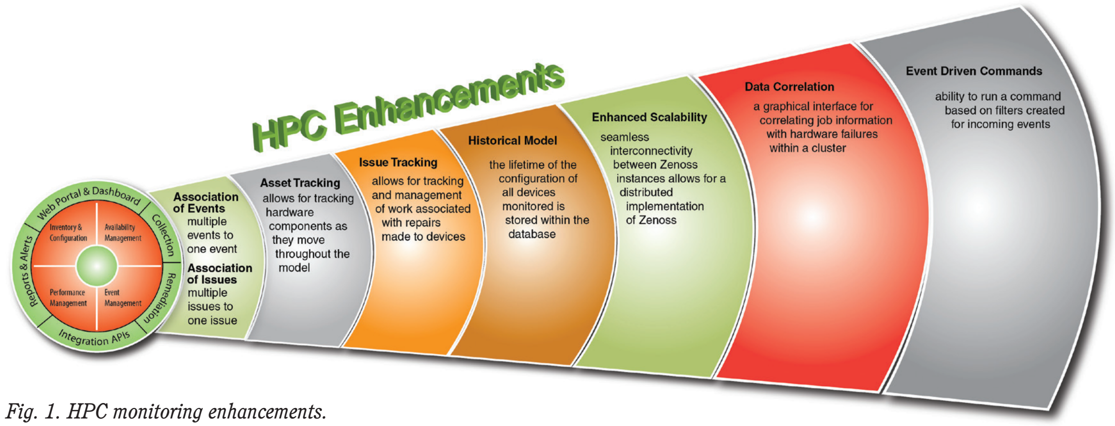

Fig. 1. HPC monitoring enhancements.

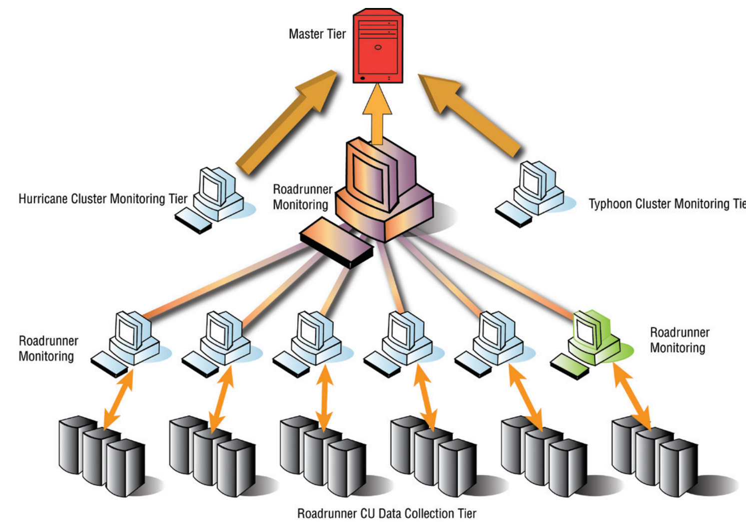

Fig. 2. HPC monitoring hierarchical implementation.

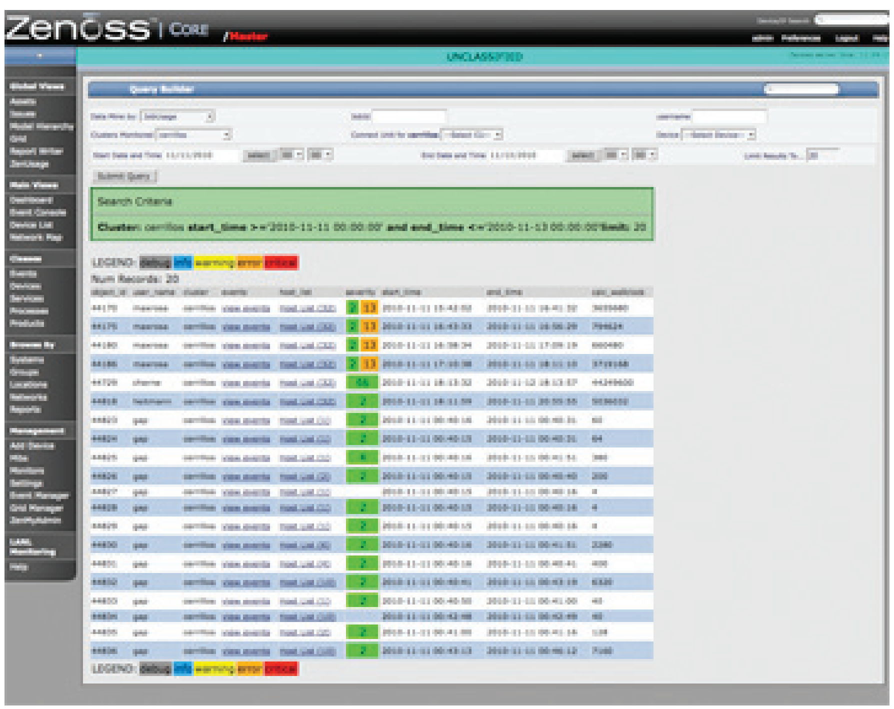

Fig. 3. Job and system event correlation interface.

Funding Acknowledgments
ASC Program